CLAIMS

What is claimed is:

1. A method for maintaining a secure operating system run-time environment comprising:

enforcing sensitivity labels such that the operating system restricts the transfer of data between subjects and objects associated with inconsistent sensitivity labels;

designating the sensitivity labels such that each sensitivity label either dominates, is dominated by, or is incomparable to each other sensitivity label; and

defining arbitrary relationships between the subjects and objects of differing sensitivity labels, thereby providing discrete access between arbitrary, incomparable sensitivity labels.

2. The method for maintaining a secure run-time environment as claimed in claim 1, wherein defining arbitrary relationships between the subjects and objects of differing sensitivity labels comprises:

labeling with sensitivity labels, all subjects and objects including, but not limited to: network connections, file system objects, processes, shared memory, and message Inter-Process Communications (IPC).

3. The method for maintaining a secure run-time environment as claimed in claim 2, wherein labeling all subjects and objects comprises listing within an encoding file a non-ambiguous sensitivity label consisting of a tag value and a label definition.

4. The method for maintaining a secure run-time environment as claimed in claim 3, wherein listing within an encoding file comprises defining hierarchical classifications of the operating system.

5. The method for maintaining a secure run-time environment as claimed in claim 3, further comprising:

providing for IPC mechanisms restriction if one of the following conditions are met:

attribute controls are mapped with privilege and an access check for effective privilege by the operating system is successful; or

attribute controls are mapped without privilege; or

an attribute Accept or Receive control is mapped and an appropriate network privilege access check is successful.

6. The method for maintaining a secure run-time environment as claimed in claim 5, wherein the IPC mechanisms comprise one or more of message queues, semaphores and shared memory.

7. The method for maintaining a secure run-time environment as claimed in claim 6, wherein an IPC channel is established if one of the following conditions are met:

the attribute controls are mapped in both directions between compartments with privilege and the access check for effective privilege by the system is successful;

the attribute controls are mapped in both directions between compartments without privilege; or

the attribute Accept or Receive control is mapped in both directions and the appropriate network privilege access check is successful.

8. The method for maintaining a secure run-time environment as claimed in claim 1, wherein defining arbitrary relationships between the subjects and objects of differing sensitivity labels comprises associating with a subject a first, arbitrary sensitivity label, associating with an object a second, arbitrary and incomparable sensitivity label, and mapping a relationship to define access between the subject and object.

9. The method for maintaining a secure run-time environment as claimed in claim 8, wherein mapping comprises:

providing discrete mandatory access control based separation between compartments that hold network interfaces and compartments that hold application processes; and

ascertaining which labels currently exist in the system at the time of a verification request as well as any labels that are scheduled to be activated at the next reboot.

10. The method for maintaining a secure run-time environment as claimed in claim 9, further comprising committing user defined configurations and verifying if the labels and controls are in effect at the time of the commit or at a next reboot, wherein the verify and commit operations return values or messages that report success and failure conditions.

11. The method for maintaining a secure run-time environment as claimed in claim 10, wherein committing user defined configurations comprises:

interfacing with an encodings file and with compartment mapping information on a real-time basis;

committing label creation and retirement or disablement operations,

mapping file and communications operations; and

verifying label creation and retirement or disablement operations to verify the file and communications mappings.

12. A secure operating system in which sensitivity labels, each comprised of a security level and one or more compartments, are enforced such that the operating system restricts the transfer of data between subjects and objects of differing sensitivity labels where a sensitivity label must dominate or be considered incomparable to other sensitivity labels comprising:

a label encodings file comprising:

a classification section to define the hierarchical names of the system in which classifications are ranked hierarchically according to an assigned level from lower to higher, wherein the higher classifications dominate the lower classifications,

a compartment section to define subdivisions of data possible within a classification,

a label section to define valid labels and their tag values, and

a communications section defining allowed communications channels between privileged and non-privileged processes that do not possess the same sensitivity label, wherein one or more channels are arbitrarily defined.

13. A secure operating system as claimed in claim 12, further comprising means for committing user defined configurations and verifying if labels and controls are in effect at a time of a commit or at a next reboot, wherein the verify and commit operations return values and messages that report success and failure conditions.

14. A secure operating system as claimed in claim 12, further comprising a network communications filter for providing TCP and UDP mapping between objects and subjects based on effective privilege, Mandatory Access Control (MAC) equivalence, mappings assigned between compartments.

15. A secure operating system as claimed in claim 14, wherein each of the mappings is with or without privilege.

16. A secure operating system as claimed in claim 12, the label encodings file further comprising:

a mapping section that governs IPC and file control mappings, wherein the communications section is operable such that a compartment to compartment communications mapping can be configured to perform an effective privilege access check by mapping attributes "with privilege" and to bypass the access check by mapping attributes "without privilege".

17. A secure operating system as claimed in claim 16, the communications section validating mapping without privilege as follows:

System Outside (attributes) ==> Vault Web Server

Vault Web CGI (attributes) ==> Vault Inside_NIC

System Outside (attributes) ==> Vault App Front End

Vault App Front End (attributes) ==> Vault App Back End

Vault App Back End (attributes) ==> Vault Inside_NIC,

and further comprising a communications decision cache, wherein whenever a MAC daemon reads and activates contents of the label encodings file, a decision list is loaded into a kernel and held in the communications decision cache.

18. A secure operating system as claimed in claim 12, wherein an entry specifies two valid sensitivity labels separated by a reserved token, "==>" (without privilege) or "P==>" (with privilege), and wherein each of the valid sensitivity labels specified may be represented by either a tag value associated with the label or an ASCII external representation of the label.

19. A secure operating system as claimed in claim 12, further comprising:

a configuration section with an additional check for allowed compartment communications and file access controls, wherein the configuration section is included in the encodings file and governs an additional check, and extensions to a MAC daemon load a configuration from the configuration section into a kernel decision cache, and an additional kernel Application Program Interface (API) is added to consult the decision cache.

20. A secure operating system as claimed in claim 12, the label section further comprising means for defining the valid labels of the system and their corresponding tag values as label entries, wherein each label entry comprises a tag value and a label definition, and wherein each defined label begins with a valid classification name.

21. A secure operating system as claimed in claim 20, wherein a classification name is followed by one or more compartment names.

22. A secure operating system as claimed in claim 12, the label section further comprising the following valid labels:

CMW_Compatible,

100: Vault Inside_NIC,

101: Vault Outside NIC,

102: Content Web Server,

103: Vault Web CGI,

104: Content Web CGI,

105: Content Web CGI,

106: Vault App Front End App Data,

107: Vault App Back End App Data,

108: Content App Data, and

109: RESERVED Content App Rdata.

23. A secure operating system as claimed in claim 12, further comprising means for determining labels that currently exist in the system at the time of a verification request as well as any labels that are scheduled to be activated at the next reboot.

24. A method for providing discrete access control between entities associated with sensitivity labels, each label comprising both a classification level component and a compartment component, comprising:

defining a fixed set of classifications for each entity;

defining a dynamic set of compartments for each entity;

partitioning application process entities and network interface entities into unique compartments; and

permitting arbitrary relationships for allowing discrete access between entities of differing sensitivity labels.

25. The method for discrete access control as claimed in claim 24, wherein the arbitrary relationships are user-defined.

26. The method for providing discrete access control as claimed in claim 24, further comprising mediating attempted security-related transactions by comparing a first sensitivity label associated with a first entity and a second sensitivity label associated with a second entity and permitting or disallowing a transaction based upon label dominance and user-defined relationships.

27. The method for providing discrete access control as claimed in claim 24, wherein defining a dynamic set of compartments for each entity creates complete information separation between Virtual Vault components, network interfaces, application content, and deployed application components.

28. The method for providing discrete access control as claimed in claim 24 further comprising providing Mandatory Access Control separation between compartments that hold network interface entities and compartments that hold application process entities.

29. The method for providing discrete access control as claimed in claim 28, wherein providing Mandatory Access Control separation between compartments that hold network interface entities and compartments that hold application process entities comprises providing discrete access controls between vanilla applications and critical network interface resources.